ANL-7634

Argonne National Laboratory

C:WRITE, A Reentrant Routine to Convert Hexadecimal Numbers to EBCDIC Decimal

by

Conrad E. Thalmayer

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Atomic Energy Commission, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona	Kansas State University	The Ohio State University
Carnegie-Mellon University	The University of Kansas	Ohio University
Case Western Reserve University	Loyola University	The Pennsylvania State University
The University of Chicago	Marquette University	Purdue University
University of Cincinnati	Michigan State University	Saint Louis University
Illinois Institute of Technology	The University of Michigan	Southern Illinois University
University of Illinois	University of Minnesota	University of Texas
Indiana University	University of Missouri	Washington University
Iowa State University	Northwestern University	Wayne State University
The University of Iowa	University of Notre Dame	The University of Wisconsin

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65

ANL-7634
Mathematics and
Computers

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

C:WRITE, A Reentrant Routine to Convert Hexadecimal Numbers to EBCDIC Decimal

by

Conrad E. Thalmayer

Chemistry Division

November 1969

PREFACE

This report describes a conversion routine for the Sigma 5 or Sigma 7 computer with Floating-Point Option. It is written in graded format, to be useful to readers of all levels of interest and sophistication: the general reader, for example, may profitably read the first one or two sections; the casual programmer will want to understand the second and third sections; only a programmer with special requirements will have need for the details of the fourth section, the flow charts, and the program listing.

This routine is independent of the computer operating system. It was written in XDS SYMBOL in October 1968 and October 1969.

TABLE OF CONTENTS

C:WRITE, A Reentrant Routine to Convert Hexadecimal Numbers to EBCDIC Decimal

by

Conrad E. Thalmayer

ABSTRACT

This report describes a reentrant, general-purpose routine for the Xerox Data Systems Sigma 5 or Sigma 7 computer with Floating-Point Option. C:WRITE converts hexadecimal numbers of the forms used in the computer into EBCDIC decimal numbers of desired length in I, E, or F format. The report explains the need for the routine, describes its capabilities, presents all the information necessary for using it, and outlines its structure. The flow charts and listing are included.

I. THE PROBLEM

In Sigma computers, numbers are hexadecimal. Let us represent the hexadecimal digits, or "higits," as 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F and indicate a hexadecimal number by X'...'. Then, for example, the number X'1A' is equal to $(1 \cdot 16^1) + (10 \cdot 16^0) = 26$. Normally, numbers are of either word (8 higit) or doubleword (16 higit) length and either fixed-point or floating-point: a fixed-point number, necessarily integral, is equal to the sum of its higits, each successive higit leftward having been multiplied by a successively higher power of 16; a floating-point number consists of a two-higit exponent followed by a 6 (or 14) higit fraction.

Outside computers, numbers are (1) normally decimal, (2) of variable length, and (3) in several formats, i.e., integral, or with point, or with point and exponent. Furthermore, (4) they are read from the computer in EBCDIC (Extended Binary Coded Decimal Interchange Code); in this code, each character is represented by a two-digit number, e.g., '1' is represented as X'F1' and 'E' is X'C5'.

For output from the computer, a routine is necessary to convert numbers of the former types into the latter. The routine should be (1) rapid, (2) brief, (3) versatile enough to satisfy the needs of all programs using it, (4) convenient to use, (5) capable of yielding output in standard format, and (6) able to recognize user errors and act appropriately. Most importantly, (7) the routine must be reentrant, i.e., while it is being used by a program of given priority it must be interruptible by one of higher priority and

subsequently resumable at the point of interruption; there should be no limit to the number of programs which might thus be sequentially interrupted while using the routine.

II. GENERAL

C:WRITE satisfies the above requirements. It accepts short (8 higit) and long (16 higit), fixed-point and floating-point numbers, and converts to EBCDIC decimal numbers of any desired length in I, F, or E format.

The seven additional requirements listed above are abetted by, inter alia, the following: (1) This routine carries out only instructions pertinent to its specific task. It does not employ subroutines. (2) The twelve tasks are written as overlapping pairs which use data in common. (3) The given number and the converted number may be at any location. (4) Only the minimum number of registers is employed, leaving the rest available to the user. (5) I format output is right-adjusted. (6) The routine will reject a request if the specified output field is too short. (7) The vital requirement of reentrancy is attained by carrying out all operations in the computer registers. Upon interruption of a program, the contents of these registers and the address of the interruption are stored in that program's Program Description Table (PDT); upon return to the program, the registers are restored and execution is resumed at the interrupted instruction. This technique relieves the user of supplying some of his working space to the routine. Inasmuch as probably every real-time program will use this routine, this will result in a major saving of core space.

III. EXTERNAL ORGANIZATION

C:WRITE has twelve entry points, bearing labels of the form xxxWRITE. The first letter of the label, L or S, indicates whether the given number is long or short; the second letter, F or I, indicates whether it is floating-point or integer (fixed-point); the third letter, I, E, or F, indicates the format of the converted number. The register utilization is as follows:

R0--User's return address
R1--Word address of given number
R2--Byte address of output field
R3--Byte address of end of output field
R4--Byte address of decimal point (F format only)

Thus, for example, if the user branches to LFIWRITE, the word whose address is in R1, together with the following word, will be treated as a long

floating-point number; the resultant integral EBCDIC decimal number, preceded by as many blanks as the field length permits, will be returned to the byte address given in R2.

The only registers altered by this routine are R1-5 in SII conversion, R1-6 in SIE, LFI, SFI, LII, LIF, SIF, and R1-7 in LFE, SFE, LIE, LFF, and SFF.

If the value given in R3 delimits a field of sufficient length, the conversion will be performed and the Condition Code set to 0. If the output field is of insufficient length, the routine will abort to the address in R0 and the Condition Code will be set to 1. For xxI conversion, the field must be long enough to contain the entire number. For xxF, there must be room for at least the integral portion of the number and the decimal point. For xxE conversion, at least three spaces must be allowed, yielding the exponent Exx; a negative exponent requires one more byte. In all three cases, a negative number requires one additional space for the sign.

IV. INTERNAL ORGANIZATION

C:WRITE consists of six pairs of routines sharing a data pool. Within each pair, either (a) the short given number is extended and treated as long or (b) the low-order half of the long number is evaluated by the Sxx routine. In each routine the result is developed one byte at a time, but not strictly left-to-right.

In SFEWRITE, the given number is loaded into R4, and R5 is cleared. If the given number is positive, (R4,5) now has the configuration of a long floating-point number and the routine branches to LFEWRITE. If the given number is negative, '-' is put into the output field specified in R2, (R2) is incremented by 1, (R4) is complemented to give (R4,5) the appropriate configuration, and the routine branches to LFEWRITE.

In LFEWRITE, the given number is loaded into R4,5; if it is negative, it is complemented, '-' is put into the output field, and (R2) is incremented. The routine may now be considered in two parts. In Part 1 the number is repeatedly multiplied by .1 or 10 until the product lies between 1 and 10; the number of these multiplications yields the decimal exponent. The routine now aborts if there is insufficient space for the exponent; otherwise the exponent is put into the right end of the output field and (R3) is set to the end of the mantissa field. In Part 2 the units digit is copied, converted to EBCDIC, and put into the output field, followed by '.'. In the rest of Part 2, which is iterated for each digit, the number is converted to fixed-point, the units digit is removed, the remainder is multiplied by 10, and the new units digit is copied, converted, and put into the output field. (R2) is incremented by 1 as each byte is developed; if (R2) is then equal to (R3), the routine exits normally.

In LIEWRITE, the given number is first compared with values, TENP, of successively smaller powers of 10, found in a table. When a value of TENP is found that is smaller than the number, it is repeatedly subtracted from the number until the number is less than TENP. This is then repeated for successive values of TENP down to 10^9, after which the routine transfers to SIEWRITE. The original value of TENP determines the exponent, to be later converted by SIEWRITE, and the number of subtractions determines each digit, which is converted and placed in the output field immediately.

In SIEWRITE, the procedure is as in LIEWRITE, but using "word" rather than "doubleword" instructions and using division rather than repeated subtraction to develop each digit. The exponent, whose value may have been determined in LIEWRITE, is finally converted to EBCDIC and placed in the output field.

In SFIWRITE, the given number is loaded into R4, 0 is loaded into R5, and the output field is cleared to blanks. If the number is negative, that is recorded, space is made for the sign, and the number is complemented. The routine then branches to LFIWRITE.

In LFIWRITE, the given number is loaded into R4,5 and the output field is cleared to blanks. If the number is negative, that is recorded, space is made for the sign, and the number is complemented. The given number is now repeatedly multiplied by .1 until its value is less than 10; at each multiplication the starting output address, originally (R3), is decreased one byte. If this value is then lower than (R2), the routine aborts. If the value of the number is less than 1, it is now set equal to 0. If it had been found to be negative, - is put into the output field. The units digit is now removed from the number, converted to EBCDIC, and put into the output field. The remainder of the number is converted to fixed-point, multiplied piecemeal by 10, and the cycle is repeated.

In LIIWRITE, the given number is loaded into R4,5, the output field is cleared to blanks, and the starting output address is set to (R3)-8. If the number is negative, that is recorded, the starting output address is decremented by 1, and the number is complemented. The given number is now compared with tabulated powers of 10, from 10^{18} to 10^9; if it is smaller than any of these, the routine branches to SIIWRITE. Otherwise, the starting output address is moved left appropriately; if it is lower than (R2), the routine aborts. If the number is negative, the sign is now put into the output field. The result is then developed by repeated subtraction from the given number of the power of 10 found above; the number of subtractions yields the digit, which is converted to EBCDIC and put into the output field; this cycle is repeated with values down to 10^9, after which the routine transfers to SIIWRITE.

In SIIWRITE, the procedure is similar to that in LIIWRITE. The main difference is that the result is developed by division of the given number by powers of ten, followed by repeated division of the remainder.

In SFFWRITE, the given number is loaded into R4, 0 is loaded into R5, and the integer portion of the output field is cleared to blanks. If the number is negative, that is recorded, space is made for the sign, and the number is complemented. The routine then branches to LFFWRITE.

In LFFWRITE, the given number is loaded into R4,5 and the integer portion of the output field is cleared to blanks. If the number is negative, that is recorded, space is made for the sign, and the number is complemented. If the given number is not less than 1, it is repeatedly multiplied by .1 until its value is less than 10; at each multiplication the starting output address, originally the units position, is decreased one byte. If this value is then lower than the given starting address, the routine aborts. If the given number has been found to be negative, - is now put into the output field. If the number is less than 1, a point is put into the output field; then the number is repeatedly multiplied by 10, and 0 is put into the output field, until either the number is no longer less than 1 or the field is filled. In the conversion loop proper, the number consists of a units digit and a fraction; the units digit is removed, converted to EBCDIC, and put into the output field; the fraction is converted to fixed-point, multiplied piecemeal by 10 to yield a new units digit, and the cycle is repeated. When the integer portion of the output field has been filled, a point is inserted and the cycle resumes until the field is filled.

In LIFWRITE, the address of LIIWRITE is stored, and the routine branches to SIFWRITE.

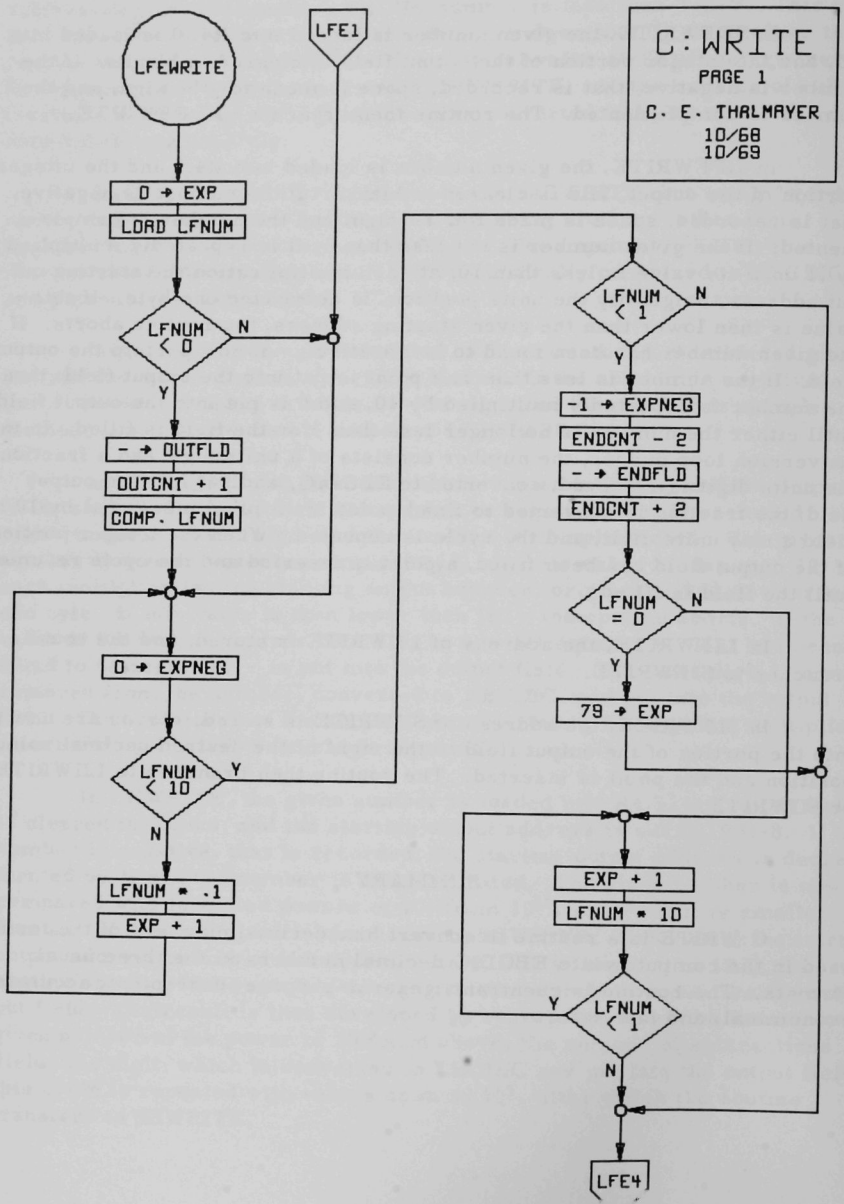
In SIFWRITE, the address of SIIWRITE is stored. Zeros are now put into the portion of the output field to the right of the desired decimal point position and the point is inserted. The routine then branches to LIIWRITE or SIIWRITE.

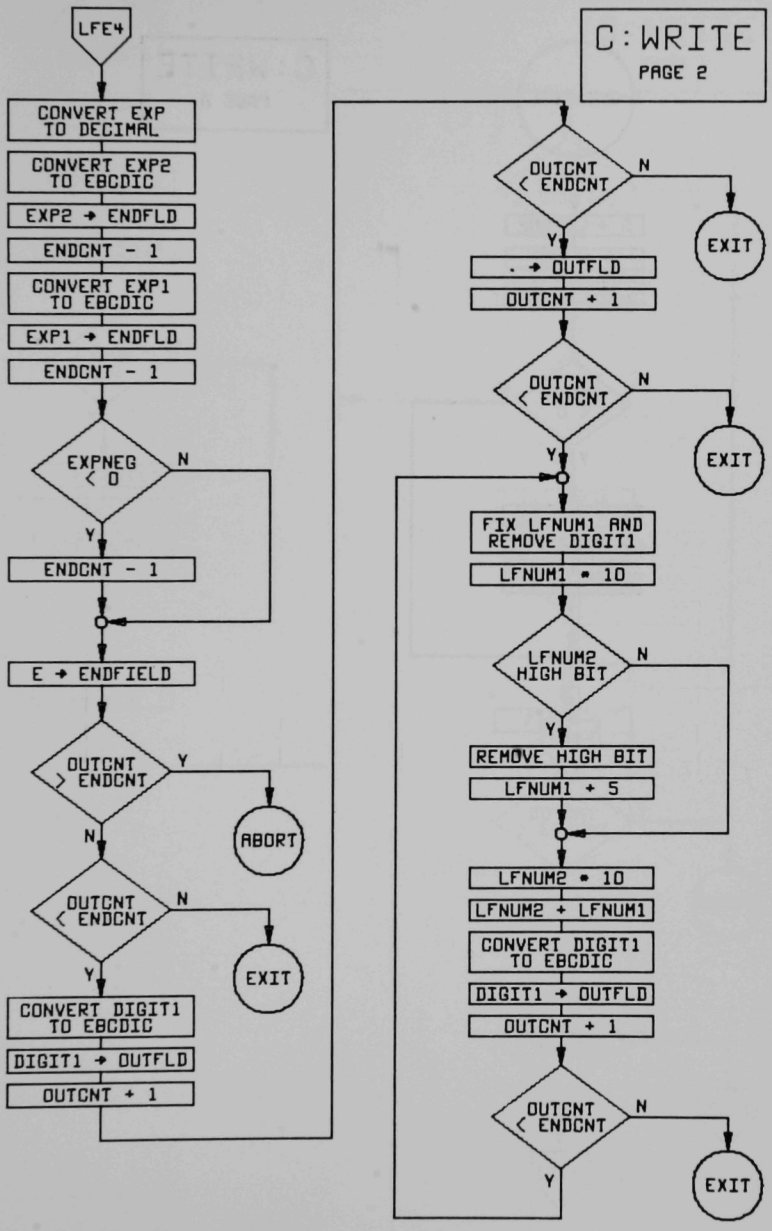
SUMMARY

C:WRITE is a routine to convert hexadecimal numbers of the forms used in the computer into EBCDIC decimal numbers in the three usual formats. The routine is reentrant, general-purpose, convenient, accurate, economical, and fail-safe.

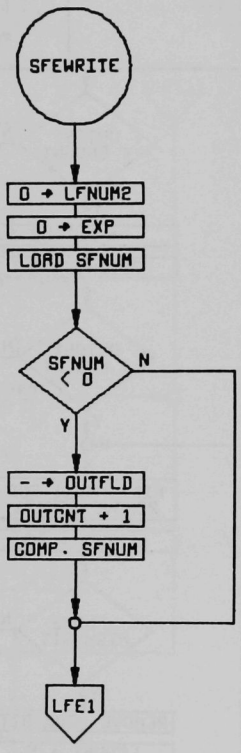
APPENDIX A

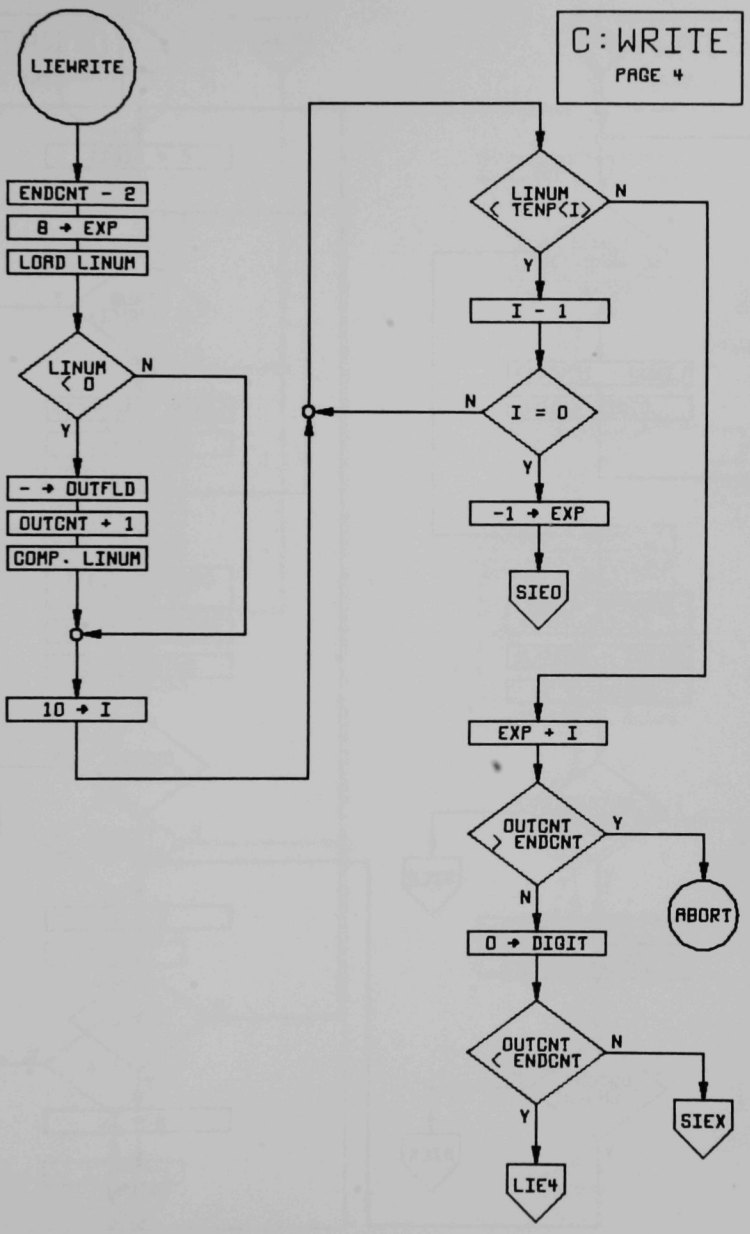
Flow Charts

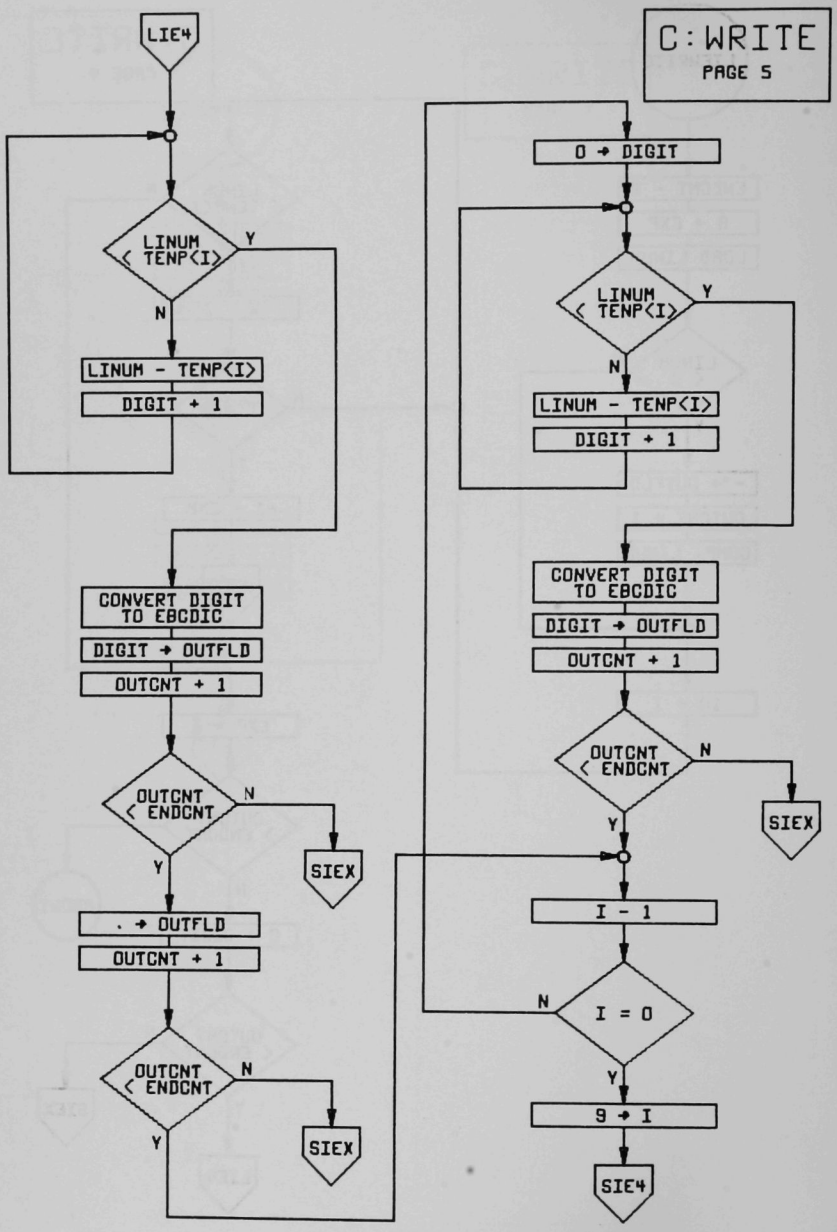


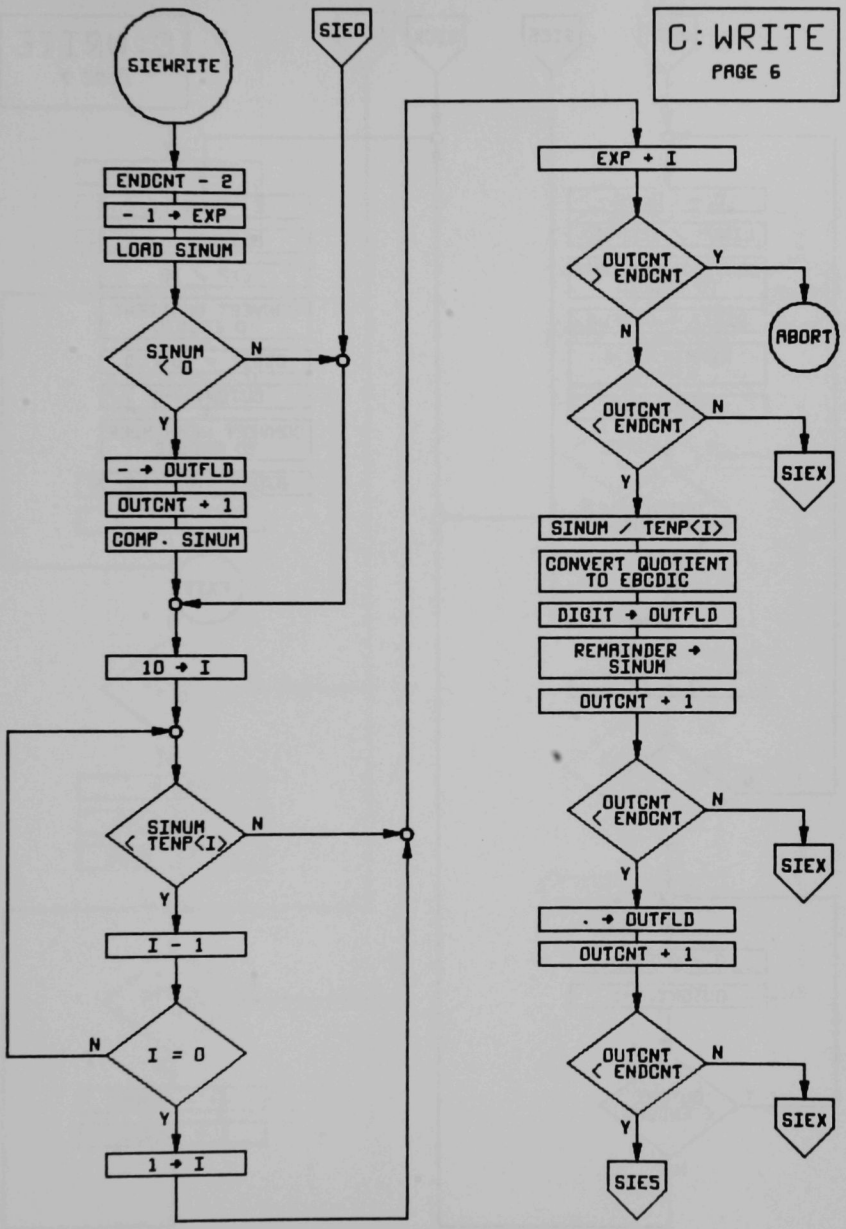

LFE4
CONVERT EXP TO DECIMAL
CONVERT EXP2 TO EBCDIC
EXP2 → ENDFLD
ENDCNT - 1
CONVERT EXP1 TO EBCDIC
EXP1 → ENDFLD
ENDCNT - 1
EXPNEG < 0
N
Y
ENDCNT - 1
E → ENDFIELD
OUTCNT > ENDCNT
Y
ABORT
N
OUTCNT < ENDCNT
N
EXIT
Y
CONVERT DIGIT1 TO EBCDIC
DIGIT1 → OUTFLD
OUTCNT + 1
C:WRITE
PAGE 2
OUTCNT < ENDCNT
N
EXIT
Y
. → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
EXIT
Y
FIX LFNUM1 AND REMOVE DIGIT1
LFNUM1 * 10
LFNUM2 HIGH BIT
N
Y
REMOVE HIGH BIT
LFNUM1 + 5
LFNUM2 * 10
LFNUM2 + LFNUM1
CONVERT DIGIT1 TO EBCDIC
DIGIT1 → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
EXIT
Y

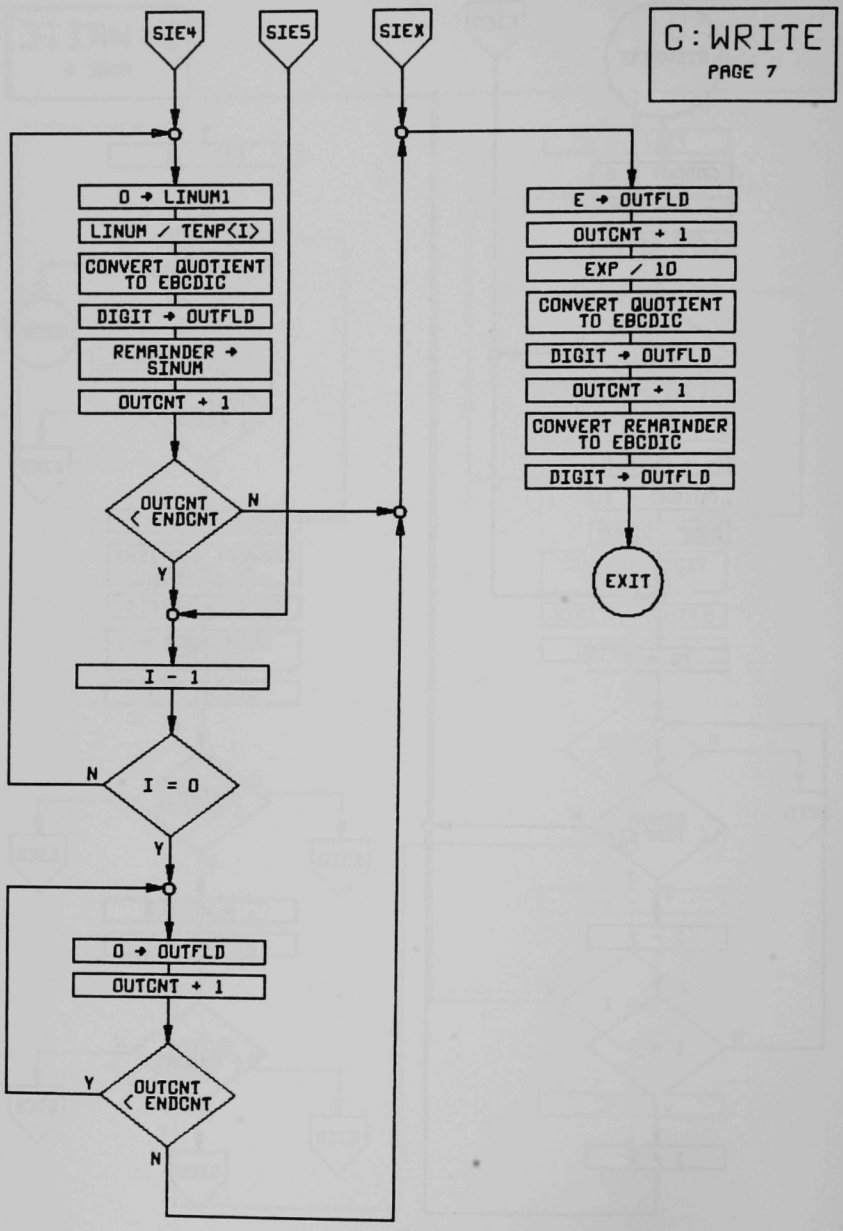
C:WRITE
PAGE 3

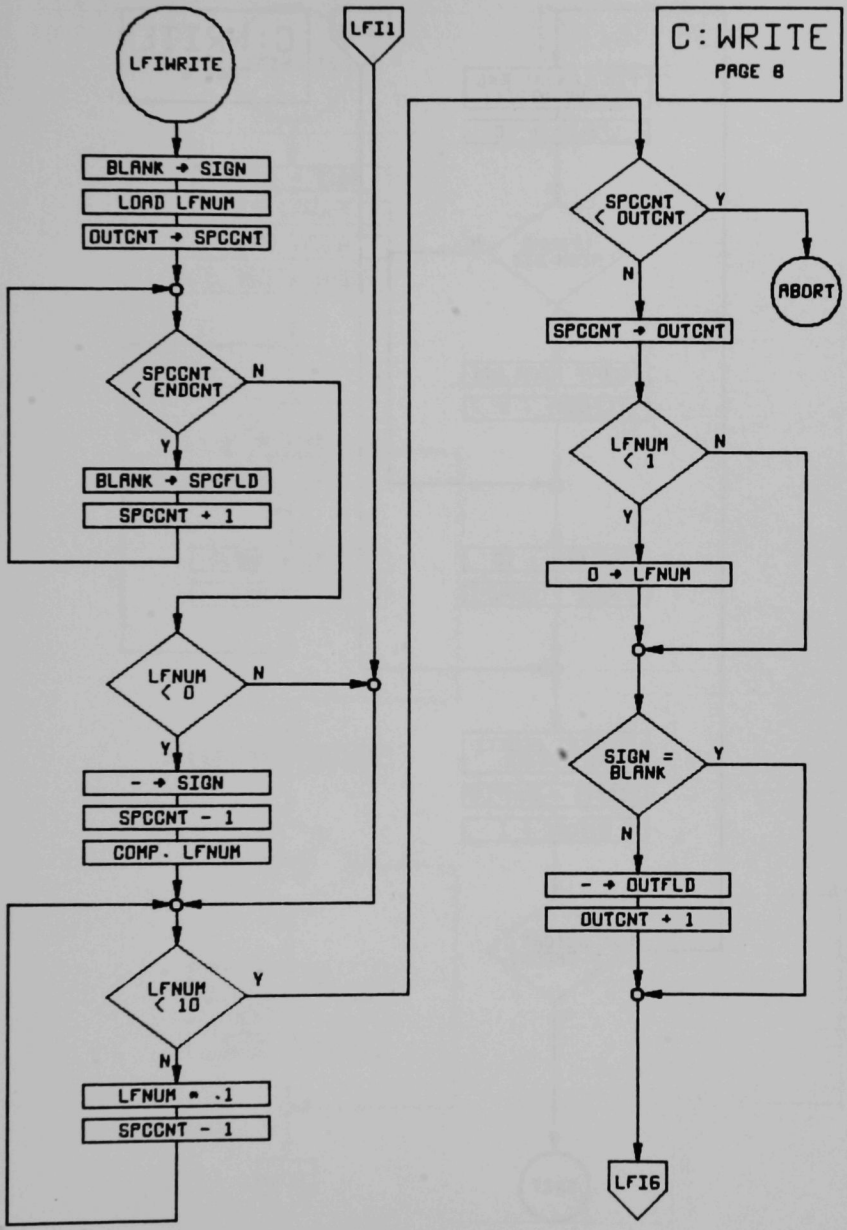


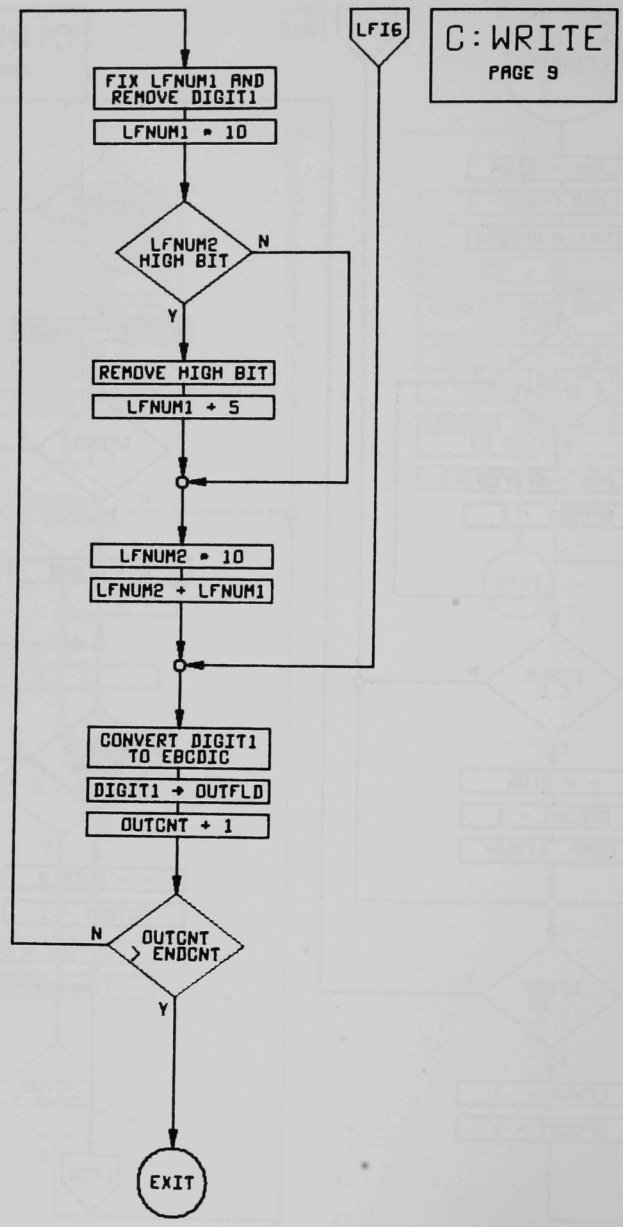

C:WRITE
PAGE 4
LIEWRITE
ENDCNT - 2
8 → EXP
LOAD LINUM
LINUM < 0
N
Y
- → OUTFLD
OUTCNT + 1
COMP. LINUM
10 → I
LINUM < TEMP<I>
N
Y
I - 1
I = 0
N
Y
-1 → EXP
SIEO
EXP + I
OUTCNT > ENDCNT
Y
N
ABORT
0 → DIGIT
OUTCNT < ENDCNT
N
Y
SIEX
LIE4

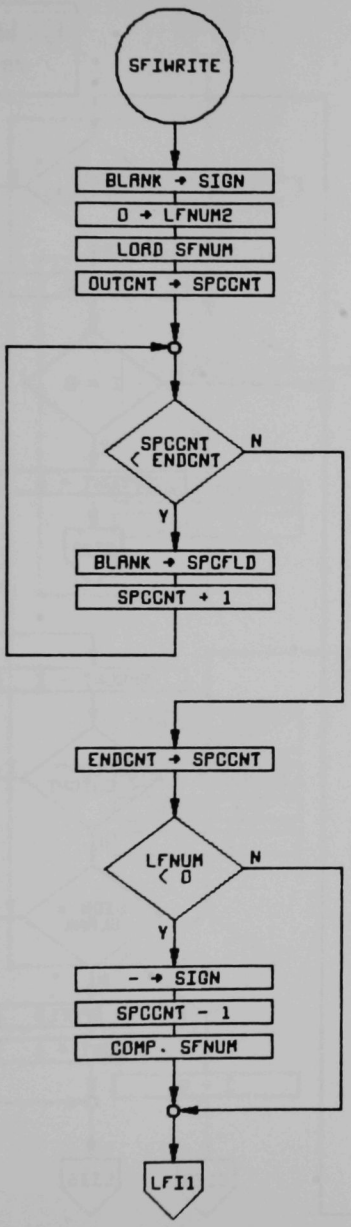

C:WRITE
PAGE 5
LIE4
LINUM < TENP<I>
Y
N
LINUM - TENP<I>
DIGIT + 1
CONVERT DIGIT TO EBCDIC
DIGIT → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
SIEX
Y
. → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
SIEX
Y
0 → DIGIT
LINUM < TENP<I>
Y
N
LINUM - TENP<I>
DIGIT + 1
CONVERT DIGIT TO EBCDIC
DIGIT → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
SIEX
Y
I - 1
I = 0
N
Y
9 → I
SIE4

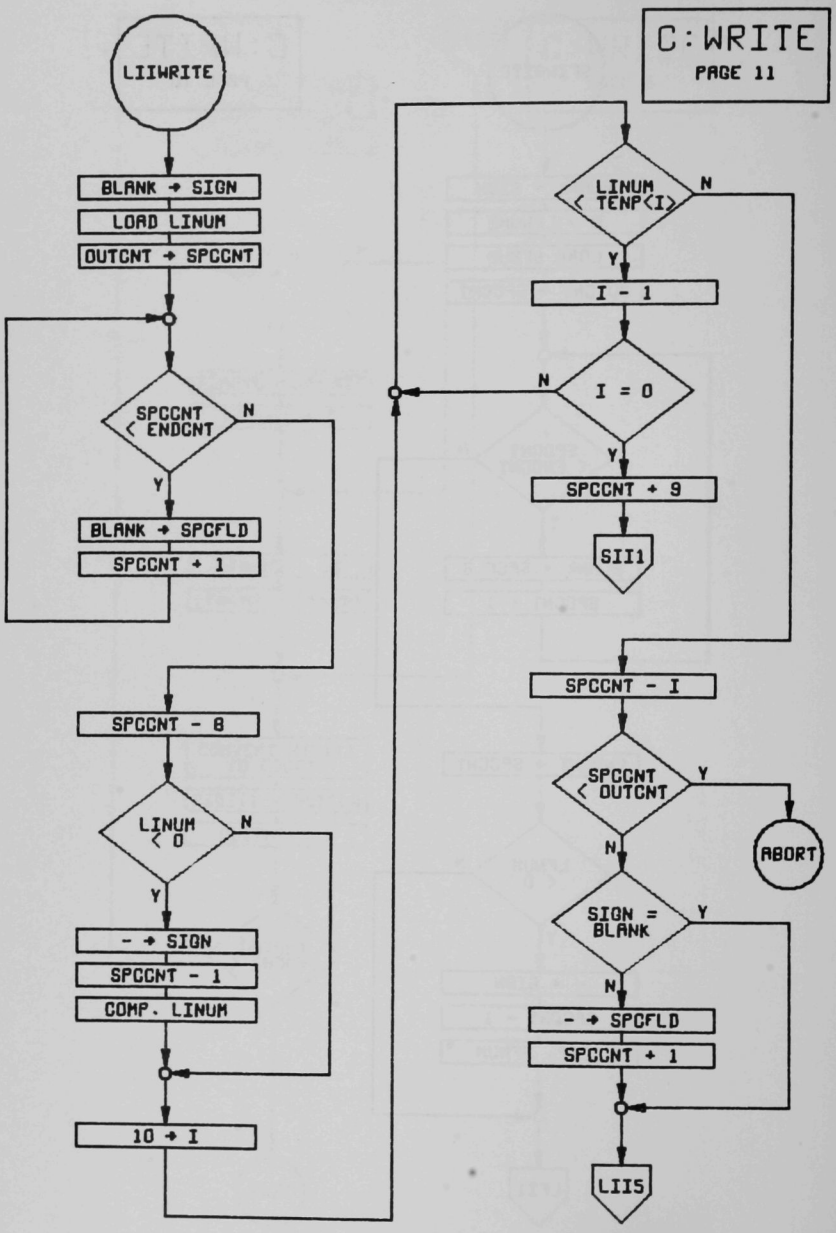

C:WRITE
PAGE 6
SIEWRITE
SIEO
ENDCNT - 2
- 1 → EXP
LOAD SINUM
SINUM < 0
N
Y
- → OUTFLD
OUTCNT + 1
COMP. SINUM
10 → I
SINUM < TENP<I>
N
Y
I - 1
I = 0
N
Y
1 → I
EXP + I
OUTCNT > ENDCNT
Y
N
ABORT
OUTCNT < ENDCNT
N
Y
SIEX
SINUM / TENP<I>
CONVERT QUOTIENT TO EBCDIC
DIGIT → OUTFLD
REMAINDER → SINUM
OUTCNT + 1
OUTCNT < ENDCNT
N
Y
SIEX
. → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
N
Y
SIEX
SIE5

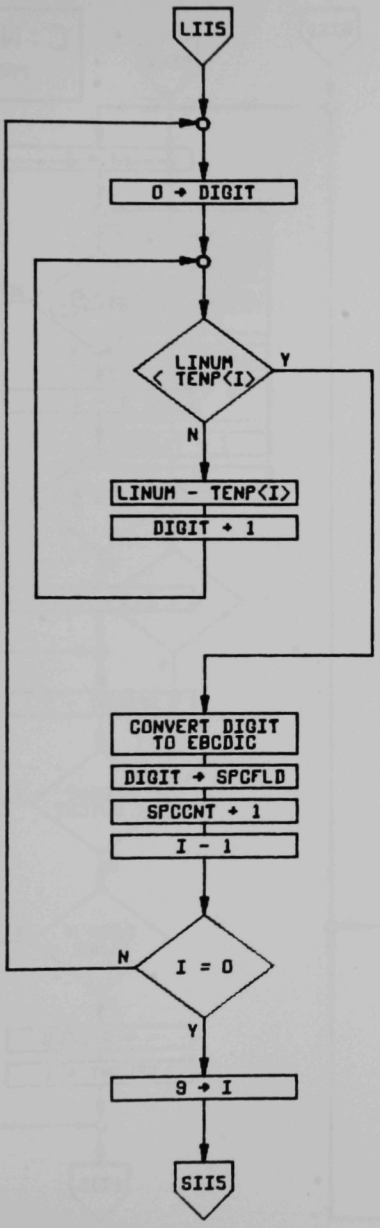

C:WRITE
PAGE 7
SIE4
SIE5
SIEX
0 → LINUM1
LINUM / TENP<I>
CONVERT QUOTIENT TO EBCDIC
DIGIT → OUTFLD
REMAINDER → SINUM
OUTCNT + 1
OUTCNT < ENDCNT
N
Y
I - 1
I = 0
N
Y
0 → OUTFLD
OUTCNT + 1
OUTCNT < ENDCNT
Y
N
E → OUTFLD
OUTCNT + 1
EXP / 10
CONVERT QUOTIENT TO EBCDIC
DIGIT → OUTFLD
OUTCNT + 1
CONVERT REMAINDER TO EBCDIC
DIGIT → OUTFLD
EXIT


C:WRITE
PAGE 8
LFIWRITE
LFI1
BLANK → SIGN
LOAD LFNUM
OUTCNT → SPCCNT
SPCCNT < ENDCNT
N
Y
BLANK → SPCFLD
SPCCNT + 1
LFNUM < 0
N
Y
- → SIGN
SPCCNT - 1
COMP. LFNUM
LFNUM < 10
Y
N
LFNUM * .1
SPCCNT - 1
SPCCNT < OUTCNT
Y
N
ABORT
SPCCNT → OUTCNT
LFNUM < 1
N
Y
0 → LFNUM
SIGN = BLANK
Y
N
- → OUTFLD
OUTCNT + 1
LFI6

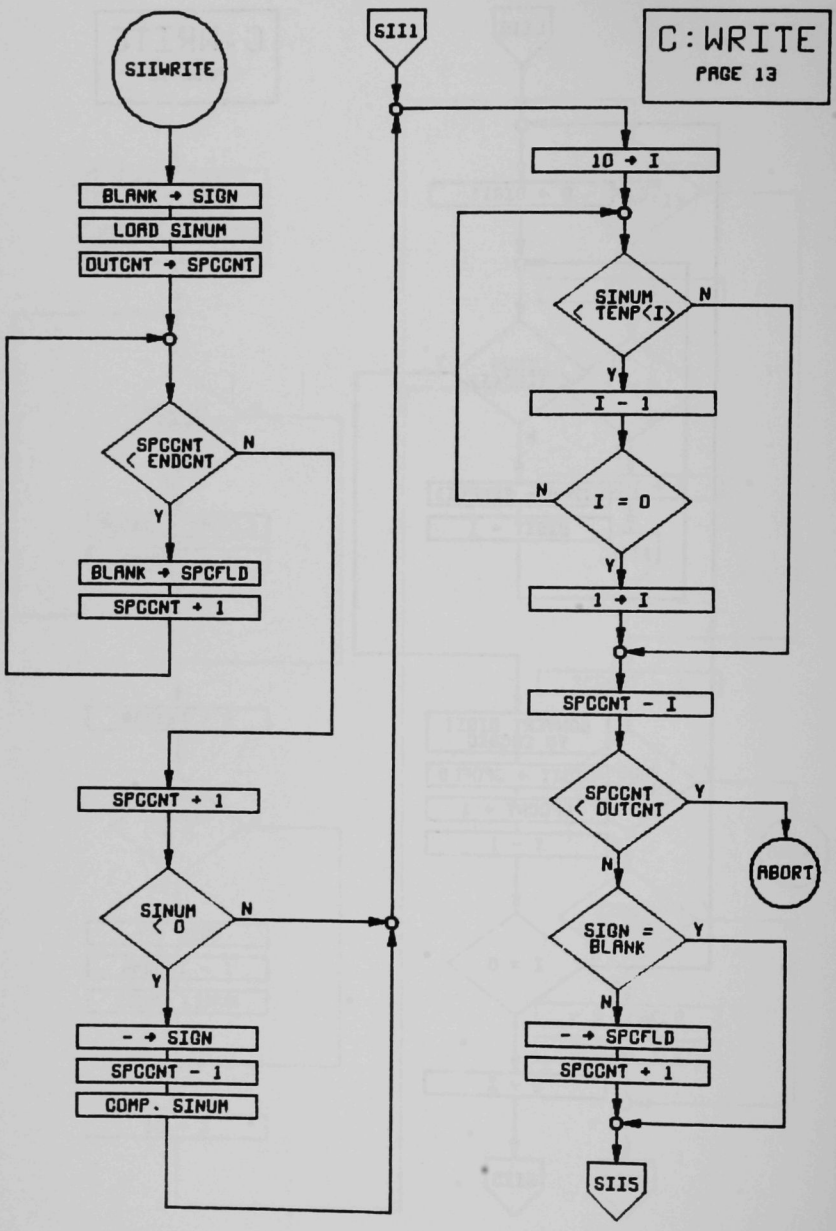

C:WRITE
PAGE 9
LFIG
FIX LFNUM1 AND REMOVE DIGIT1
LFNUM1 * 10
LFNUM2 HIGH BIT
N
Y
REMOVE HIGH BIT
LFNUM1 + 5
LFNUM2 * 10
LFNUM2 + LFNUM1
CONVERT DIGIT1 TO EBCDIC
DIGIT1 → OUTFLD
OUTCNT + 1
N
OUTCNT > ENDCNT
Y
EXIT

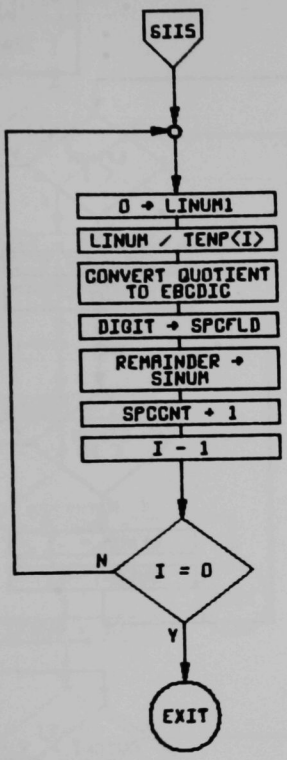

C:WRITE
PAGE 10
SFIWRITE
BLANK → SIGN
0 → LFNUM2
LOAD SFNUM
OUTCNT → SPCCNT
SPCCNT < ENDCNT
N
Y
BLANK → SPCFLD
SPCCNT + 1
ENDCNT → SPCCNT
LFNUM < 0
N
Y
- → SIGN
SPCCNT - 1
COMP. SFNUM
LFI1


C:WRITE
PAGE 11
LIIWRITE
BLANK → SIGN
LOAD LINUM
OUTCNT → SPCCNT
SPCCNT < ENDCNT
N
Y
BLANK → SPCFLD
SPCCNT + 1
SPCCNT - 8
LINUM < 0
N
Y
- → SIGN
SPCCNT - 1
COMP. LINUM
10 → I
LINUM < TEMP<I>
N
Y
I - 1
I = 0
N
Y
SPCCNT + 9
SII1
SPCCNT - I
SPCCNT < OUTCNT
Y
N
ABORT
SIGN = BLANK
Y
N
- → SPCFLD
SPCCNT + 1
LII5

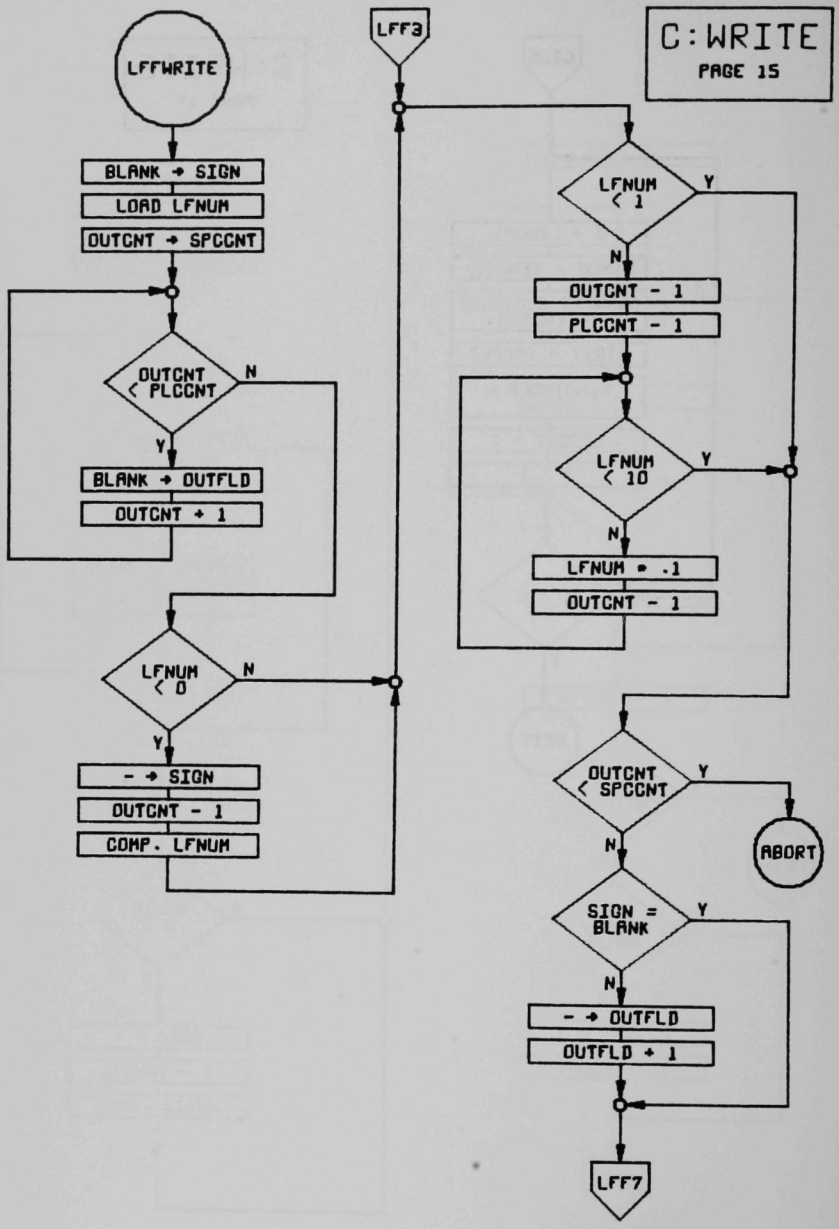


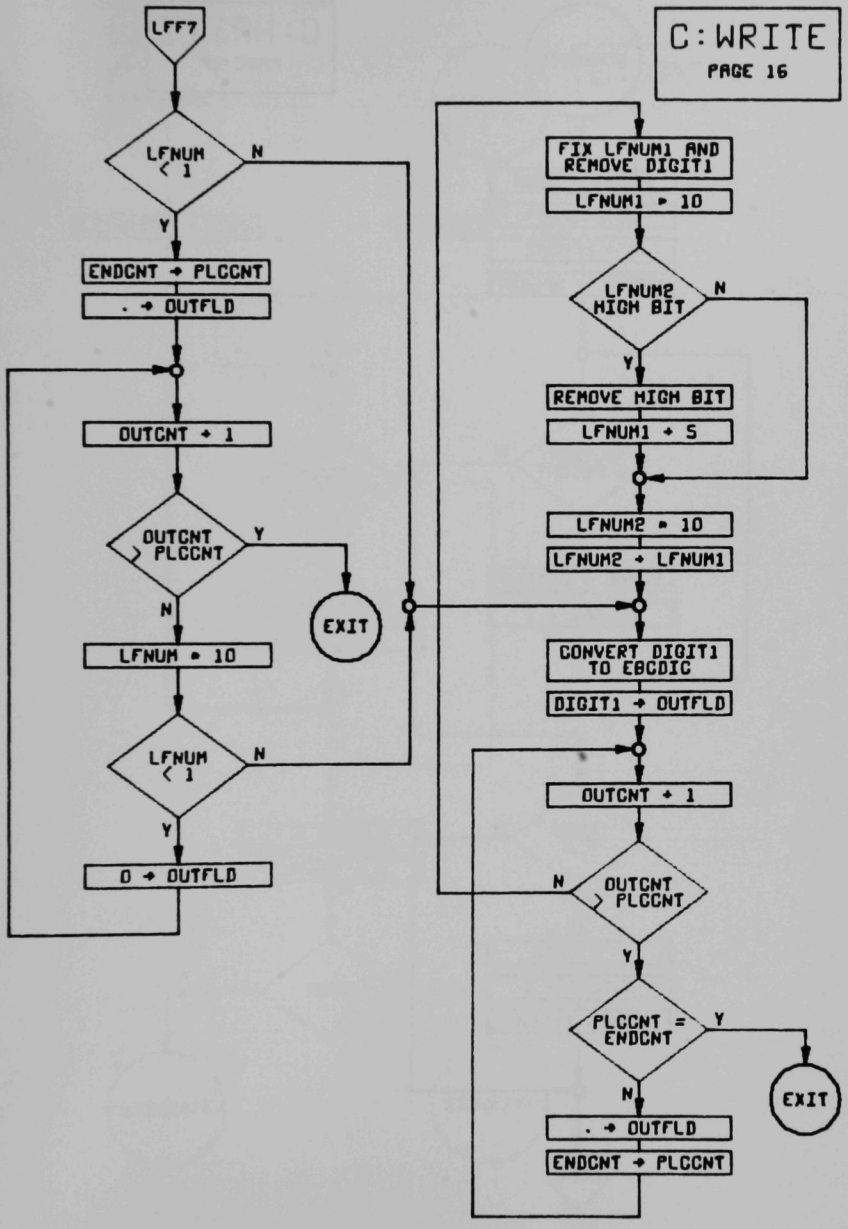
C:WRITE
PAGE 12

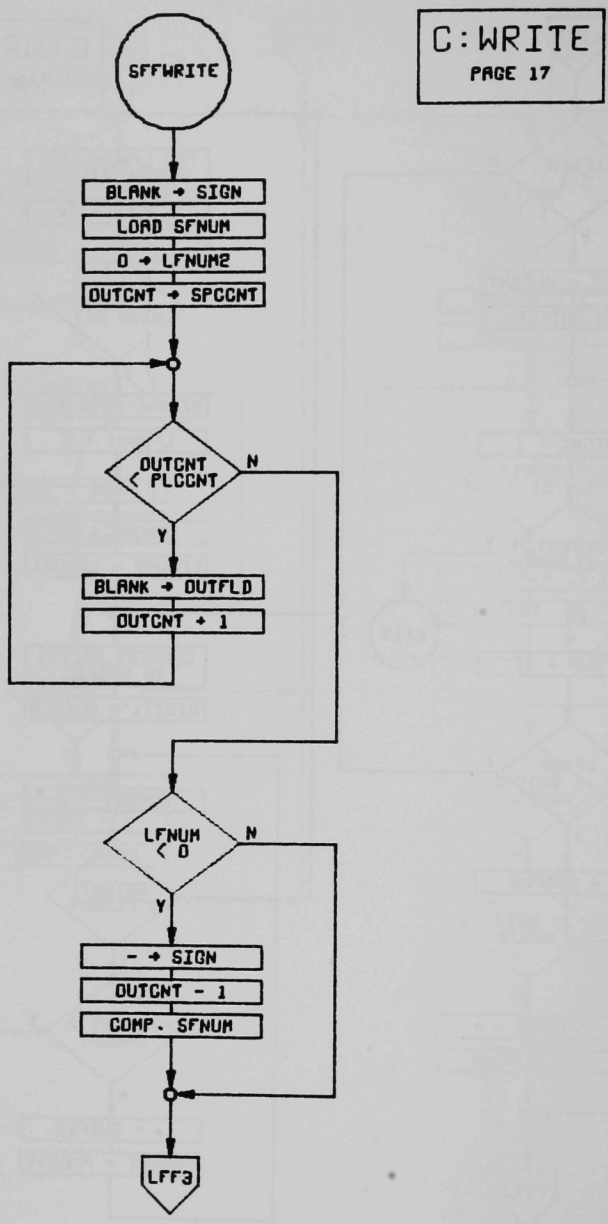

SIIWRITE
BLANK → SIGN
LOAD SINUM
OUTCNT → SPCCNT
SPCCNT < ENDCNT
N
Y
BLANK → SPCFLD
SPCCNT + 1
SPCCNT + 1
SINUM < 0
N
Y
- → SIGN
SPCCNT - 1
COMP. SINUM
SII1
10 → I
SINUM < TENP<I>
N
Y
I - 1
I = 0
N
Y
1 → I
SPCCNT - I
SPCCNT < OUTCNT
Y
N
ABORT
SIGN = BLANK
Y
N
- → SPCFLD
SPCCNT + 1
SII5
C:WRITE
PAGE 13

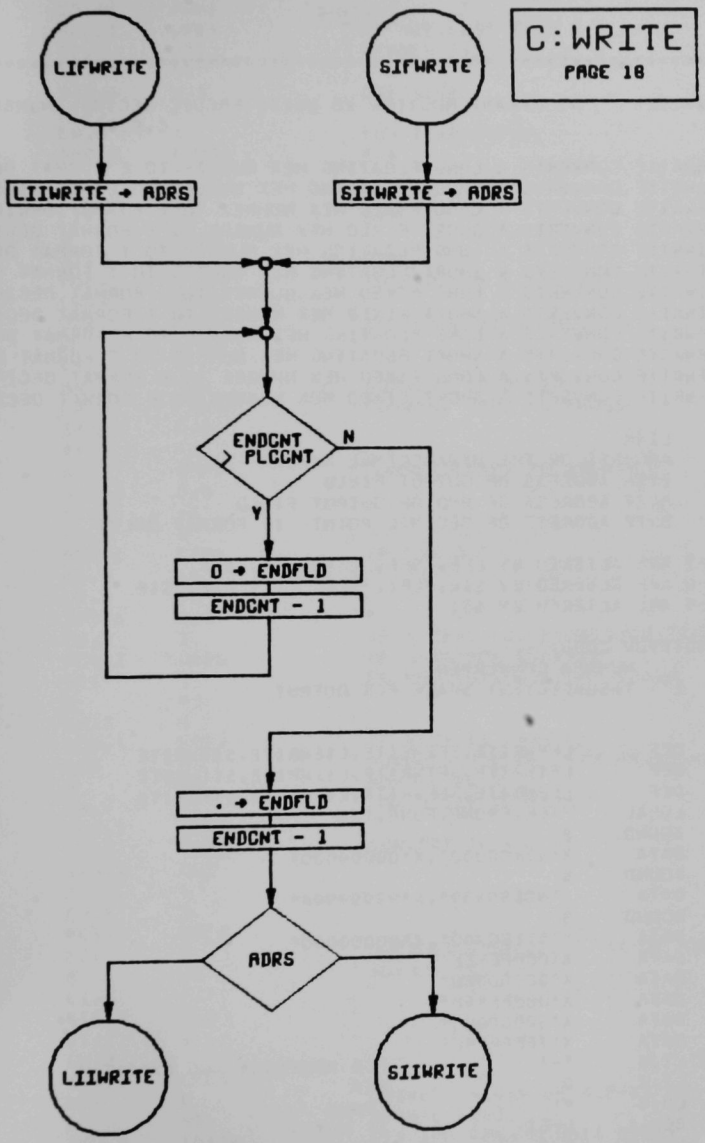


C:WRITE
PAGE 14


C:WRITE
PAGE 15
LFFWRITE
BLANK → SIGN
LOAD LFNUM
OUTCNT → SPCCNT
OUTCNT < PLCCNT
N
Y
BLANK → OUTFLD
OUTCNT + 1
LFNUM < 0
N
Y
- → SIGN
OUTCNT - 1
COMP. LFNUM
LFF3
LFNUM < 1
Y
N
OUTCNT - 1
PLCCNT - 1
LFNUM < 10
Y
N
LFNUM * .1
OUTCNT - 1
OUTCNT < SPCCNT
Y
N
ABORT
SIGN = BLANK
Y
N
- → OUTFLD
OUTFLD + 1
LFF7


C:WRITE
PAGE 16
LFF7
LFNUM < 1
N
Y
ENDCNT → PLCCNT
. → OUTFLD
OUTCNT + 1
OUTCNT > PLCCNT
Y
N
EXIT
LFNUM * 10
LFNUM < 1
N
Y
0 → OUTFLD
FIX LFNUM1 AND REMOVE DIGIT1
LFNUM1 * 10
LFNUM2 HIGH BIT
N
Y
REMOVE HIGH BIT
LFNUM1 + 5
LFNUM2 * 10
LFNUM2 + LFNUM1
CONVERT DIGIT1 TO EBCDIC
DIGIT1 → OUTFLD
OUTCNT + 1
OUTCNT > PLCCNT
N
Y
PLCCNT = ENDCNT
Y
EXIT
N
. → OUTFLD
ENDCNT → PLCCNT


C:WRITE
PAGE 17
SFFWRITE
BLANK → SIGN
LOAD SFNUM
0 → LFNUM2
OUTCNT → SPCCNT
OUTCNT < PLCCNT
N
Y
BLANK → OUTFLD
OUTCNT + 1
LFNUM < 0
N
Y
- → SIGN
OUTCNT - 1
COMP. SFNUM
LFF3


LIFWRITE
SIFWRITE
C:WRITE
PAGE 18
LIIWRITE → ADRS
SIIWRITE → ADRS
ENDCNT > PLCCNT
N
Y
0 → ENDFLD
ENDCNT - 1
. → ENDFLD
ENDCNT - 1
ADRS
LIIWRITE
SIIWRITE

APPENDIX B

Listing

```
*************************************************************************
*
***  C:WRITE      REENTRANT ROUTINE TO WRITE EBCDIC DECIMAL NUMBERS
*                                                    C.E.T.        10/17/69
*
*     LFEWRITE CONVERTS A LONG FLOATING HEX NUMBER TO E FORMAT DECIMAL
*     SFEWRITE CONVERTS A SHORT FLOATING HEX NUMBER TO E FORMAT DECIMAL
*     LIEWRITE CONVERTS A LONG FIXED HEX NUMBER TO E FORMAT DECIMAL
*     SIEWRITE CONVERTS A SHORT FIXED HEX NUMBER TO E FORMAT DECIMAL
*     LFIWRITE CONVERTS A LONG FLOATING HEX NUMBER TO I FORMAT DECIMAL
*     SFIWRITE CONVERTS A SHORT FLOATING HEX NUMBER TO I FORMAT DECIMAL
*     LIIWRITE CONVERTS A LONG FIXED HEX NUMBER TO I FORMAT DECIMAL
*     SIIWRITE CONVERTS A SHORT FIXED HEX NUMBER TO I FORMAT DECIMAL
*     LFFWRITE CONVERTS A LONG FLOATING HEX NUMBER TO F FORMAT DECIMAL
*     SFFWRITE CONVERTS A SHORT FLOATING HEX NUMBER TO F FORMAT DECIMAL
*     LIFWRITE CONVERTS A LONG FIXED HEX NUMBER TO F FORMAT DECIMAL
*     SIFWRITE CONVERTS A SHORT FIXED HEX NUMBER TO F FORMAT DECIMAL
*
*     R0   LINK
*     R1   ADDRESS OF THE HEXADECIMAL NUMBER
*     R2   BYTE ADDRESS OF OUTPUT FIELD
*     R3   BYTE ADDRESS OF END OF OUTPUT FIELD
*     R4   BYTE ADDRESS OF DECIMAL POINT  (F FORMAT ONLY)
*
*     R1-7 ARE ALTERED BY LFE, SFE, LIE, LFF, SFF
*     R1-6 ARE ALTERED BY SIE, LFI, SFI, LII, LIF, SIF
*     R1-5 ARE ALTERED BY SII
*
*     CONDITION CODE:
*         0   NUMBER CONVERTED
*         1   INSUFFICIENT SPACE FOR OUTPUT
*
*
          DEF       LFEWRITE,SFEWRITE,LIEWRITE,SIEWRITE
          DEF       LFIWRITE,SFIWRITE,LIIWRITE,SIIWRITE
          DEF       LFFWRITE,SFFWRITE,LIFWRITE,SIFWRITE
          LOCAL     FTEN,FPONE,FONE,TEN
          BOUND     8
FTEN      DATA      X'41A00000',X'00000000'
          BOUND     8
FPONE     DATA      X'40199999',X'9999999A'
          BOUND     8
FONE      DATA      X'41100000',X'00000000'
EMASK     DATA      X'00FFFFFF'
FMASK     DATA      X'000000F0'
IMASK     DATA      X'000FFFFF'
HMASK     DATA      X'80000000'
XMASK     DATA      X'7FFFFFFF'
LFEWRITE  LI,6      '-'             FOR MANTISSA AND EXPONENT
          LI,7      0               EXP
          LD,4      *1              LFNUM
          BCR,1     LFE1            IF NOT -
          STB,6     0,2             MANTISSA SIGN
          AI,2      1               ADDRESS OF NEXT BYTE
          LCD,4     4               MAKE LFNUM +
LFE1      LI,1      0               EXPNEG
          CD,4      FTEN
          BCS,1     LFE2            IF LESS THAN 10
          FML,4     FPONE           TO PRODUCE X.XXXXXX
```

```
        AI,7    1               INCREMENT EXP
        B       LFE1
LFE2    CD,4    FONE
        BCR,1   LFE4            IF NOT LESS THAN 1
        LI,1    -1              EXPNEG SIGNAL
        AI,3    -2
        STB,6   0,3             EXP SIGN
        AI,3    2
        LW,6    4               FOR ZERO CHECK
        AND,6   EMASK           FIX
        CI,6    0
        BCS,3   LFE3            IF NOT 0
        LI,7    79              EXP
        B       LFE4
LFE3    AI,7    1               EXP
        FML,4   FTEN            TO PRODUCE X.XXXXXXXXXXXXX
        CD,4    FONE
        BCR,1   LFE4            IF NOT LESS THAN 1
        B       LFE3
LFE4    LI,6    0
        DW,6    TEN             CONVERT TO DECIMAL
        OR,6    FMASK           CONVERT EXP2 TO EBCDIC
        STB,6   0,3
        AI,3    -1
        AI,7    X'F0'           CONVERT EXP1 TO EBCDIC
        STB,7   0,3
        AI,3    -1
        CI,1    0
        BCR,1   LFE5            IF EXP NOT -
        AI,3    -1
LFE5    LI,6    'E'
        STB,6   0,3
        CW,2    3               NEXT BYTE VS. FORBIDDEN SPACE
        BCR,2   LFE6            IF NEXT NOT GREATER
        LCI     1               ERROR: INSUFFICIENT SPACE
        B       *0
LFE6    CW,2    3
        BCS,1   LFE7            IF NEXT IS SMALLER
        LCI     0               OUTPUT: EXX OR E-XX OR -E-XX
        B       *0              NORMAL EXIT
LFE7    LW,7    4               FOR DIGIT COPYING
        SLS,7   -20
        OR,7    FMASK           CONVERT TO EBCDIC
        STB,7   0,2
        AI,2    1
        CW,2    3
        BCS,1   LFE8
        LCI     0               OUTPUT: XEXX OR XE-XX OR -XE-XX
        B       *0              NORMAL EXIT
LFE8    LI,7    '.'
        STB,7   0,2
        AI,2    1
        CW,2    3
        BCS,1   LFE9
        LCI     0               OUTPUT: X.EXX OR X.E-XX OR -X.E-XX
        B       *0              NORMAL EXIT
LFE9    AND,4   IMASK           FIX AND REMOVE UNIT DIGIT
        LW,7    4               LFNUM1
        MI,7    10              LFNUM1*10
```

```
          CW,5     HMASK          SIGN BIT
          BCR,4    LFEA           IF ABSENT
          AND,5    XMASK          REMOVE BIT
          AI,7     5              A*8
LFEA      MI,4     10             LFNUM2*10
          AW,4     7              LFNUM*10
          LW,7     4              FOR DIGIT COPYING
          SLS,7    -20
          OR,7     FMASK          CONVERT TO EBCDIC
          STB,7    0,2
          AI,2     1
          CW,2     3
          BCS,1    LFE9
          LCI      0              NORMAL EXIT
          B        *0             (END OF LFEWRITE)
SFEWRITE  LI,5     0              TO EXTEND SFNUM TO LFNUM
          LI,6     '-'            FOR MANTISSA AND EXPONENT
          LI,7     0              EXP
          LW,4     *1             SFNUM
          BCR,1    LFE1           IF NOT -: TREAT AS LFNUM
          STB,6    0,2            MANTISSA SIGN
          AI,2     1              ADDRESS OF NEXT BYTE
          LCW,4    4              MAKE SFNUM +
          B        LFE1           TREAT AS LFNUM  (END OF SFEWRITE)
          BOUND    8
TEN09     DATA,8   1000000000
TEN10     DATA     X'00000002',X'540BE400'
TEN11     DATA     X'00000017',X'4876E800'
TEN12     DATA     X'000000E8',X'D4A51000'
TEN13     DATA     X'00000918',X'4E72A000'
TEN14     DATA     X'00005AF3',X'107A4000'
TEN15     DATA     X'00038D7E',X'A4C68000'
TEN16     DATA     X'002386F2',X'6FC10000'
TEN17     DATA     X'01634578',X'5D8A0000'
TEN18     DATA     X'0DE0B6B3',X'A7640000'
LIEWRITE  AI,3     -2             0,3 IS FIRST BYTE FORBIDDEN TO MANTISSA
          LI,6     8              EXPONENT
          LD,4     *1
          BCR,1    LIE0           IF NOT -
          LI,7     '-'
          STB,7    0,2
          AI,2     1
          LCD,4    4
LIE0      LI,1     10             INDEX FOR TEN POWERS (=EXP-8)
LIE1      CD,4     TEN09-1,1
          BCR,1    LIE2
          BDR,1    LIE1
          LI,6     -1             FOR SIEWRITE
          B        SIE0           FOR TREATMENT AS SINGLE WORD
LIE2      AW,6     1              EXPONENT
          CW,2     3
          BCR,2    LIE3
          LCI      1              ERROR: INSUFFICIENT SPACE
          B        *0             ABORT
LIE3      LI,7     0              DIGIT
          CW,2     3
          BCR,1    SIEX           WRITE EXP ONLY
LIE4      CD,4     TEN09-1,1
          BCS,1    LIE5
```

```
          SD,4      TEN09-1,1
          AI,7      1
          B         LIE4
LIE5      AI,7      X'F0'          EBCDIC DIGIT
          STB,7     0,2
          AI,2      1
          CW,2      3
          BCR,1     SIEX
          LI,7      '.'
          STB,7     0,2
          AI,2      1
          CW,2      3
          BCR,1     SIEX
          B         LIE9
LIE6      LI,7      0
LIE7      CD,4      TEN09-1,1
          BCS,1     LIE8
          SD,4      TEN09-1,1
          AI,7      1
          B         LIE7
LIE8      AI,7      X'F0'
          STB,7     0,2
          AI,2      1
          CW,2      3
          BCR,1     SIEX
LIE9      BDR,1     LIE6
          LI,1      9
          B         SIE4           (END OF LIEWRITE: BRANCH TO SIEWRITE)
ONE       DATA      1
TEN       DATA      10
TEN2      DATA      100
TEN3      DATA      1000
TEN4      DATA      10000
TEN5      DATA      100000
TEN6      DATA      1000000
TEN7      DATA      10000000
TEN8      DATA      100000000
TEN9      DATA      1000000000
SIEWRITE  AI,3      -2             *3 IS FIRST BYTE FORBIDDEN TO MANTISSA
          LI,6      -1             EXPONENT
          LW,5      *1
          BCR,1     SIE0           IF NOT -
          LI,4      '-'
          STB,4     0,2
          AI,2      1
          LCW,5     5
SIE0      LI,1      10             INDEX FOR TEN POWERS (=EXP+1)
SIE1      CW,5      ONE-1,1
          BCR,1     SIE2
          BDR,1     SIE1
          LI,1      1              IF SINUM=0
SIE2      AW,6      1              EXPONENT
          CW,2      3
          BCR,2     SIE3           IF NOT BEYOND LIMIT
          LCI       1              ERROR: INSUFFICIENT SPACE
          B         *0             ABORT
SIE3      LI,4      0              QUOTIENT = DIGIT
          CW,2      3
          BCR,1     SIEX
```

```
            DW,4    ONE-1,1
            AI,5    X'F0'           EBCDIC FIRST DIGIT
            STB,5   0,2
            LW,5    4
            AI,2    1
            CW,2    3
            BCR,1   SIEX
            LI,4    '.'
            STB,4   0,2
            AI,2    1
            CW,2    3
            BCR,1   SIEX
            B       SIE5
SIE4        LI,4    0
            DW,4    ONE-1,1
            AI,5    X'F0'           EBCDIC DIGIT
            STB,5   0,2
            LW,5    4
            AI,2    1
            CW,2    3
            BCR,1   SIEX
SIE5        BDR,1   SIE4
            LI,4    '0'
SIE6        STB,4   0,2
            AI,2    1
            CW,2    3
            BCS,1   SIE6
SIEX        LI,4    'E'
            STB,4   0,2
            AI,2    1
            LI,4    0
            LW,5    6
            DW,4    TEN
            AI,5    X'F0'
            STB,5   0,2             EXPONENT FIRST DIGIT
            AI,2    1
            OR,4    FMASK
            STB,4   0,2             EXPONENT SECOND DIGIT
            LCI     0
            B       *0              (END OF SIEWRITE AND LIEWRITE)
LFIWRITE    LI,6    ' '
            LD,4    *1              LFNUM
            LW,1    2
LFI0        CW,1    3
            BCR,1   LFI7
            STB,6   0,1
            AI,1    1
            B       LFI0
LFI7        CI,4    0
            BCR,1   LFI1            IF NOT -
            LI,6    '-'             SIGN
            AI,1    -1              FOR SIGN
            LCD,4   4               MAKE LFNUM +
LFI1        CD,4    FTEN
            BCS,1   LFI2            IF LESS THAN 10
            FML,4   FPONE           TO PRODUCE X.XXXXXXXXXXXXX
            AI,1    -1              EXPAND OUTFIELD
            B       LFI1
LFI2        CW,1    2               REQUIRED VS. ALLOWED FIRST BYTE ADDRESS
```

```
         BCR,1    LFI3
         LCI      1              INSUFFICIENT SPACE
         B        *0             ABORT
LFI3     LW,2     1              ADDRESS OF NEXT OUTPUT BYTE
         CW,4     FONE
         BCR,1    LFI8
         LI,4     0
LFI8     CI,6     ' '
         BCR,3    LFI6           IF NOT -
         STB,6    0,2            SIGN
         AI,2     1              INCREMENT BYTE ADDRESS
         B        LFI6
LFI4     AND,4    IMASK          FIX AND REMOVE UNIT DIGIT
         LW,1     4              LFNUM1
         MI,1     10             LFNUM1*10
         CW,5     HMASK          SIGN BIT
         BCR,4    LFI5           IF ABSENT
         AND,5    XMASK          REMOVE BIT
         AI,1     5              A*8
LFI5     MI,4     10             LFNUM2*10
         AW,4     1              LFNUM*10
LFI6     LW,1     4              FOR DIGIT COPYING
         SLS,1    -20
         OR,1     FMASK          CONVERT TO EBCDIC
         STB,1    0,2
         AI,2     1              INCREMENT BYTE ADDRESS
         CW,2     3
         BCR,2    LFI4
         LCI      0              NORMAL EXIT
         B        *0             (END OF LFIWRITE)
SFIWRITE LI,6     ' '
         LI,5     0
         LW,4     *1             SFNUM
         LW,1     2
SFI0     CW,1     3
         BCR,1    SFI1
         STB,6    0,1
         AI,1     1
         B        SFI0
SFI1     CI,4     0
         STW,3    1              ADDRESS OF OUTPUT BYTE
         BCR,1    LFI1           IF NOT -: TREAT AS LFNUM
         LI,6     '-'            SIGN
         AI,1     -1             FOR SIGN
         LCW,4    4              MAKE SFNUM +
         B        LFI1           TREAT AS LFNUM  (END OF SFIWRITE)
LIIWRITE LI,6     ' '
         LD,4     *1             LINUM
         LW,1     2
LII0     CW,1     3
         BCR,1    LII8
         STB,6    0,1
         AI,1     1
         B        LII0
LII8     AI,3     -8
         CI,4     0
         BCR,1    LII1           IF NOT -
         LI,6     '-'
         AI,3     -1             FOR SIGN
```

```
          LCD,4     4               MAKE LINUM +
LII1      LI,1      10              INDEX FOR TEN POWERS
LII2      CD,4      TEN09-1,1       FIND NEXT LOWER TEN POWER
          BCR,1     LII3
          BDR,1     LII2
          AI,3      9               FOR SIIWRITE
          LW,4      6               FOR SIIWRITE
          B         SII1            TREAT AS SINUM
LII3      SW,3      1               FIRST BYTE ADDRESS
          CW,3      2               VS. ALLOWED FIRST ADDRESS
          BCR,1     LII4
          LCI       1               INSUFFICIENT SPACE
          B         *0              ABORT
LII4      CI,6      ' '
          BCR,3     LII5            IF NOT -
          STB,6     0,3             SIGN
          AI,3      1
LII5      LI,6      0               DIGIT
LII6      CD,4      TEN09-1,1
          BCS,1     LII7
          SD,4      TEN09-1,1
          AI,6      1
          B         LII6
LII7      AI,6      X'F0'           EBCDIC DIGIT
          STB,6     0,3
          AI,3      1
          BDR,1     LII5
          LI,1      9               FOR SIIWRITE
          B         SII5            TREAT AS SINUM  (END OF LIIWRITE)
SIIWRITE  LI,4      ' '
          LW,5      *1              SINUM
          LW,1      2
SII0      CW,1      3
          BCR,1     SII6
          STB,4     0,1
          AI,1      1
          B         SII0
SII6      AI,3      1
          CI,5      0
          BCR,1     SII1            IF NOT -
          LI,4      '-'
          AI,3      -1              FOR SIGN
          LCW,5     5               MAKE SINUM +
SII1      LI,1      10              INDEX FOR TEN POWERS
SII2      CW,5      ONE-1,1         FIND NEXT LOWER TEN POWER
          BCR,1     SII3
          BDR,1     SII2
          LI,1      1               FOR ZERO
SII3      SW,3      1               STARTING BYTE ADDRESS
          CW,3      2               VS. ALLOWED STARTING ADDRESS
          BCR,1     SII4
          LCI       1               INSUFFICIENT SPACE
          B         *0              ABORT
SII4      CI,4      ' '
          BCR,3     SII5            IF NOT -
          STB,4     0,3             SIGN
          AI,3      1
SII5      LI,4      0               QUOTIENT = DIGIT
          DW,4      ONE-1,1
```

```
         AI,5     X'F0'          EBCDIC DIGIT
         STB,5    0,3
         LW,5     4              REMAINDER
         AI,3     1
         BDR,1    SII5
         LCI      0
         B        *0             (END OF SIIWRITE)
LFFWRITE LI,6     ' '
         LW,7     4              POINT BYTE ADDRESS
         LD,4     *1             LFNUM
         LW,1     2              FIRST BYTE ADDRESS
LFF1     CW,2     7              OUTPUT ADDRESS VS. POINT ADDRESS
         BCR,1    LFF2
         STB,6    0,2            BLANK INTO OUTFIELD
         AI,2     1
         B        LFF1
LFF2     CI,4     0              LFNUM
         BCR,1    LFF3           IF NOT -
         LI,6     '-'            SIGN
         AI,2     -1             FOR SIGN
         LCD,4    4              MAKE LFNUM +
LFF3     CW,4     FONE
         BCS,1    LFF5           IF LESS THAN ONE
         AI,2     -1
         AI,7     -1             UNITS ADDRESS
LFF4     CD,4     FTEN
         BCS,1    LFF5           IF LESS THAN 10
         FML,4    FPONE          TO PRODUCE X.XXXXXXXXXXXXX
         AI,2     -1             EXPAND OUTFIELD
         B        LFF4
LFF5     CW,2     1              REQUIRED VS. ALLOWED FIRST BYTE ADDRESS
         BCR,1    LFF6
         LCI      1              INSUFFICIENT SPACE
         B        *0             ABORT
LFF6     CI,6     ' '
         BCR,3    LFF7           IF NOT -
         STB,6    0,2            SIGN
         AI,2     1              INCREMENT BYTE ADDRESS
         B        LFF7
LFF7     LI,6     '.'
         CW,4     FONE
         BCR,1    LFF8           IF NOT LESS THAN ONE
         LW,7     3              END ADDRESS
         STB,6    0,2            POINT
         LI,1     '0'
LFF8     AI,2     1              INCREMENT OUTFIELD
         CW,2     7              OUTFIELD VS. ENDFIELD
         BCS,2    LFFD
         FML,4    FTEN           TO PRODUCE X.XXXXXXXXXXXXX
         CW,4     FONE
         BCR,1    LFF8
         STB,1    0,2            ZERO
         B        LFF8
LFF9     AND,4    IMASK          FIX AND REMOVE UNIT DIGIT
         LW,1     4              LFNUM1
         MI,1     10             LFNUM1*10
         CW,5     HMASK          SIGN BIT
         BCR,4    LFFA           IF ABSENT
         AND,5    XMASK          REMOVE BIT
```

```
            AI,1    5               A*8
LFFA        MI,4    10              LFNUM2*10
            AW,4    1               LFNUM*10
LFFB        LW,1    4               FOR DIGIT COPYING
            SLS,1   -20
            OR,1    FMASK           CONVERT TO EBCDIC
            STB,1   0,2
LFFC        AI,2    1               INCREMENT OUTPUT ADDRESS
            CW,2    7               VS. UNITS ADDRESS OR END ADDRESS
            BCR,2   LFF9
            CW,7    3
            BCR,3   LFFD            IF END ADDRESS
            LW,7    3               END ADDRESS
            STB,6   0,2             POINT
            B       LFFC
LFFD        LCI     0               NORMAL EXIT
            B       *0              (END OF LFFWRITE)
SFFWRITE    LI,6    ' '
            LW,7    4               POINT BYTE ADDRESS
            LW,4    *1              SFNUM
            LI,5    0               FOR LFNUM
            LW,1    2               FIRST BYTE ADDRESS
SFF1        CW,2    7               OUTPUT ADDRESS VS. UNITS ADDRESS
            BCR,1   SFF2
            STB,6   0,2             BLANK INTO OUTFIELD
            AI,2    1
            B       SFF1
SFF2        CI,4    0               LFNUM
            BCR,1   LFF3            IF NOT -: TREAT AS LFNUM
            LI,6    '-'             SIGN
            AI,2    -1              FOR SIGN
            LCW,4   4               MAKE SFNUM +
            B       LFF3            TREAT AS LFNUM  (END OF SFFWRITE)
LIFWRITE    LI,5    LIIWRITE        FOR LII CONVERSION
            B       SIF1            (END OF LIFWRITE)
SIFWRITE    LI,5    SIIWRITE        FOR SII CONVERSION
SIF1        LI,6    0
SIF2        CW,3    4               END ADDRESS VS. POINT ADDRESS
            BCR,2   SIF3
            STB,6   0,3             ZERO
            AI,3    -1              DECREMENT END ADDRESS
            B       SIF2
SIF3        LI,6    '.'
            STB,6   0,3             POINT
            AI,3    -1              UNITS ADDRESS
            B       *5              (END OF SIFWRITE)
            END
```

ACKNOWLEDGMENTS

I am grateful to Paul Day and Henry Krejci for the discussions in which the approach to the problem was developed.